REMARKS

Claims 1-2, 4-41, 43-71 and 72-90 remain pending in the application. Reconsideration is respectfully requested in light of the following remarks.

Double Patenting Rejection:

The Examiner rejected claims 1, 2, 4-41, 43-71 and 73-90 under the judiciary created doctrine of obviousness-type double patenting as being unpatentable over claims 1-66 of U.S. Patent 7,200,848. A terminal disclaimer to obviate the double patenting rejection over claims 1-66 has been filed along with this response. Accordingly, Applicant respectfully requests removal of the double patenting rejection of claims 1, 2, 4-41, 43-71 and 73-90.

Section 103(a) Rejection:

The Examiner rejected claims 1, 2, 4, 6-41, 43-71 and 73-90 under 35 U.S.C. § 103(a) as being unpatentable over Johnson ("XML JavaBeans Integration, Part 3", 7/1999) in view of Allen (U.S. Patent 6,658,625), claim 5 as being unpatentable over Johnson in view of Allen, and further in view of Gillam ("Java Liaison" column, 3/1999). Applicants respectfully traverse these rejections for at least the following reasons.

Regarding claim 1, contrary to the Examiner's assertions, Johnson in Allen fails to teach or suggest **processing the first object into an intermediary hash table representation of the first object, wherein at least one entry of the intermediary hash table representation includes: a hash key including a name of an instance variable of the first object; and a value for the instance variable.** The Examiner relies on Allen, citing col. 12, lines 13-19 and col. 13, lines 33-44. However, the Examiner's reliance on Allen (even if combined with Johnson) is misplaced.

Allen teaches a system for generic data conversion to support calling various version or releases of server API from a single data description. Allen teaches parsing XML-based (e.g., PCML) data descriptions and using those descriptions to convert data, such as server API parameters, between different formats (Allen, col. 6, lines 35-47). Johnson teaches converting JavaBean objects to and from XML using an Intermediary Document Object Model (DOM) tree structure (Johnson, p. 3, para. 1-8).

The Examiner argues, "Allen teaches [that] using such hash table was known in the pertinent art" (Office Action, p. 5). Applicants respectfully disagree with the Examiner's interpretation. Allen teaches using a hash table to hold information extracted from parsing the <u>XML-based data descriptions</u>. Nowhere does Allen mention anything about *processing a <u>computer programming language object</u>* into an intermediary hash table representation, as recited in Applicants' claim. Instead, Allen specifically teaches using a hash table in a traditional manner to improve and enhance accessing parsed textual data (e.g., PCML data descriptions). For instance, Allen teaches, "The data description is preferably a text file that can be easily manipulated and changed with simple text editors" (Allen, col. 7, lines 2-7). In other words, Allen's use of a hash table is directed toward caching and quickly accessing *textual* data descriptions. (Allen, col. 11, lines 41-55 and col. 12, lines 5-17). At no point in Allen's system, even if combined with Johnson's XMLJavaBeans, is a computer programming language object processed into an intermediary hash table representation of the object.

Furthermore, Allen's hash table, even when combined with Johnson's teachings, does not include a hash key including a name of an instance variable of the object and a value for the instance variable, as recited in Applicants' claim. As noted above, Allen teaches using a hash table to stored parsed textual data descriptions – not instances of computer programming language objects. Allen's data descriptions simply do not include the values for instance variables of an object that is processed into a intermediary hash table representation of the object. Rather, Allen's data descriptions include information describing the type, size, character set, and location of data elements (Allen, col. 6, lines 35-47 and col. 7, lines 2-11).

Thus, a system resulting from the Examiner's combination of Johnson and Allen would not include processing an object into an intermediary hash table representation of the object. Instead, including Allen's hash table into Johnson's XMLJavaBeans would allow Johnson to store parsed XML data in a hash table, as taught by Johnson, but would not include processing an object into a hash table representation of that object since as shown above, neither Johnson nor Allen teaches or suggests using a hash table in that manner.

To establish a *prima facie* obviousness of a claimed invention, all claim limitations must be taught or suggested by the prior art. *In re Royka*, 490 F.2d 981, 180 U.S.P.Q. 580 (C.C.P.A. 1974), MPEP 2143.03. More specifically, "All words in a claim must be considered in judging the patentability of that claim against the prior art" *In re Wilson*, 424 f.2d 1382, 1385,165 USPQ 494, 496 (CCPA 1970). As shown above, the cited art does not teach or suggest all limitations of Applicants' claims. As such, the Examiner has failed to provide a *prima facie* rejection of claim 1.

Thus, the rejection of claim 1 is not supported by the cited art and removal thereof is respectfully requested. Similar remarks also apply to claims 40 and 71.

Regarding claim 12, Johnson in view of Allen fails to teach or suggest a decompilation process of the virtual machine generating the first object from the data representation language representation of the first object, where the first object is an instance of a class in the computer programming language and providing the first object to a second process executing within the virtual machine, contrary to the Examiner's assertions. The Examiner relies on Allen, stating, "Allen teaches [that it] is well known in the pertinent art … to reconstitute the object state in a distributed system" and citing col. 3, lines 12-24. However, the Examiner's reliance on Allen, even if combined with Johnson, is misplaced. Firstly, Applicants' claim does not recite merely reconstituting the object state in a distributed system. The Examiner is ignoring the specific language of Applicants' claim. Allen does not teach a decompilation process generating the object from the data representation language representation of the object and providing the object to a second process executing within the virtual machine, as recited in Applicants' claim. Even if Allen were to teach "reconstitute[ing] object state" as relied on by the Examiner, the Examiner's combination of cited art would still fail to teach or suggest the specific limitations of Applicants' claim, as discussed below.

Johnson teaches an XMLJavaBeans class intended to provide a single process the ability to transform a JavaBean in memory into an XML document and vice versa. Allen teaches a method for generic data conversion including an XML parser to read and validate PCML data descriptions. However, nowhere does either Johnson or Allen (singly or in combination) mention providing a data representation language representation of an object ***to a second process*** **(i.e., <u>a different process</u> than the first process from which the data representation language representation of a first computer programming language object was received)**. Johnson's XMLJavaBeans class is clearly configured to return the results (e.g., the object generated from an XML document) to the same method, and hence <u>to the same process</u>, that called the XMLJavaBeans interface (See, e.g., XMLBeanReader method definition, page 9). Allen's data converter and XML parser are directed to using PCML data descriptions to aid in the automatic conversion of data among different "multiple releases of the server using the same data description" and to "call the server program with the correct, converted parameters in the correct order" (Allen, col. 3, lines 7-24).

Allen fails to mention sending a data representation language representation of an object to a second (different) process. Allen's teaching regarding sending converter parameter data when calling server APIs does not teach or suggest the specific limitation of providing an object, generated from a data representation language representation of the object, to a second process. Allen describes generic data conversion and the use of traditional parameter data when calling server APIs. While Allen's converted data may be sent over a network to a server, Allen's data is sent across in various Server API formats, which do not include, and are not described as including, data representation language representations of object. Allen use of data representation languages (e.g., PCML) is limited to data descriptions, not representations of object. Nor does Allen teach the use of computer programming objects generated from data representation language representations of objects.

The Examiner's broad statement that Allen teaches that it was well known to "reconstitute object state" does not address the specific limitation and language of Applicants' claim.

Moreover, a server receiving Allen's converted data is clearly not a second process executing **within the virtual machine**, as recited in Applicants' claim.

Thus, the rejection of claim 12 is not supported by the cited art and removal thereof is respectfully requested. Similar remarks also apply to claim 78.

Regarding claim 25, Johnson in view of Allen fails to teach or suggest, <u>generating a message in a data representation language, where the message includes a data representation language representation of the object</u>, as recited in Applicants' claim. The Examiner relies on Allen, citing col. 3, lines 12-14 and asserting, "Allen teaches generating XML message of an object and providing such XML representation to another process" (Office Action, p. 12). Applicants respectfully disagree with the Examiner's interpretation of Allen. Allen does not, even if combined with Johnson, teach generating a message in a data representation language including a data representation language representation of the object. Instead, as noted above, Allen teaches using XML (e.g., PCML) for data descriptions, such as for descriptions of the data type, size, character set, and the location and length of the data elements within a file or other data stream (Allen, col. 6, lines 35-41). Allen uses XML to describe data formats, **<u>not</u> for messages including data representation language representations of objects**. Allen's system does not involve sending messages in a data representation language. Instead, Allen teaches using the XML data descriptions to convert data into a format suitable for a particular release of a server API. For instance, Allen describes how the XML data definitions may specify both input and output parameters for particular server functions to be called (Allen, col. 6, lines 40-47; col. 9, lines 30-47). Allen, even if combined with Johnson, simply does not teach or suggest (or even mention) generating a message in a data representation language including a data representation language representation of an object.

Furthermore, the Examiner's combination of cited art would not result in a system including generating messages in a data representation language including a data representation language representation of an object. For instance, incorporating Allen's XML data descriptions and generic data conversion into Johnson's XML JavaBeans system would include Johnson's ability to convert JavaBeans to and from XML, and would be able to use XML-based data descriptions when converting data. However, since, as shown above, neither Johnson nor Allen mentions generating a message in a data representation language or about including a data representation language representation of an object in such a message, no system resulting from a combination of Johnson and Allen would include such functionality.

Moreover, the Examiner has not provided a valid reason for modifying Johnson in view of Allen. The Examiner states that it would have been obvious to modify Johnson's system because "one having ordinary skill in the art would be motivated to reproduce object state in the distributed system" (Office Action, p. 12). However, desiring to "reproduce object state" would not motivate one to include the generic data conversion system of Allen into Johnson's XMLJavaBeans. Firstly of all, Allen is not concerned with, nor teach anything regarding, reproducing object state in a distributed system. Instead, Allen is concerned with automatic generic data conversion to support the calling of different releases of server APIs. Additionally, the Examiner's stated opinion that one "would be motivated to reproduce object state" does not provide an actual, valid, reason why one of ordinary skill would be motivated to combine Johnson and Allen. The Examiner has simply stated that one would be motivated, but fails to actually provide any reason as to why one would include the generic data conversion system of Allen into the XMLJavaBean system of Johnson. That the Examiner believes one would be motivated does not provide such a reason and is not supported by any actual evidence of record.

Thus, the rejection of claim 25 is not supported by the cited art and removal thereof is respectfully requested. Similar remarks also apply to claims 62 and 84.

Regarding claim 34, Johnson in view of Allen fails to teach or suggest <u>a first process receiving a message in a data representation language from a second process, where the message includes information representing a computer programming language object</u>. The Examiner relies on Allen, citing col. 3, lines 12-24. However, as shown above regarding the rejection of claim 25, Allen, even if combined with Johnson, does not teach or suggest message in a data representation language or including information representing a computer programming language object in such a message, contrary to the Examiner's assertions.

Instead, Allen teaches using XML-based (e.g., PCML) data descriptions to support automatic data conversions among various releases of server APIs. For instance, Allen's system reads and parses a XML-based data description that describes various data elements, such as input and output parameters to server API functions, so that the data can be converted to a suitable format for the particular release of the server API being called (Allen, col. 6, lines 35-41). Allen's data descriptions are simply not messages in a data representation language. Instead, Allen's data descriptions are stored in an XML file that is read and parsed (not send as a message) by Allen's system.

Further supporting the fact that Allen's system does not involve messages in a data representation language, is Allen's teaching regarding the use of a hash table to more quickly utilize the data descriptions (Allen, col. 12, lines 5-27). Hence, Allen teaches parses the XML-based data descriptions and then converting them into a hash table for subsequent access and not generating messages in a data representation language.

As shown above, the cited art does not teach or suggest all the limitations of Applicants' claim. Thus, the rejection of claim 34 is not supported by the cited art and removal thereof is respectfully requested. Similar remarks also apply to claim 50.

Applicants also assert that the rejection of numerous ones of the dependent claims is further unsupported by the teachings of the cited art. However, since the rejection of the independent claims has been shown to be unsupported by the cited art, a further discussion of the rejection of the dependent claims is not necessary at this time.

CONCLUSION

Applicants submit the application is in condition for allowance, and notice to that effect is respectfully requested.

If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5181-72000/RCK.

Respectfully submitted,

/Robert C. Kowert/
Robert C. Kowert, Reg. #39,255
Attorney for Applicants

Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8850

Date: November 30, 2007